三、視覺化軟體建構方法

在第二章的討論上，我們可以見到，在傳統的使用者介面開發流程與方式所會產生的問題，而本章會介紹另一個以視覺化爲導向的軟體開發方法，並由此方法再擴充，加上以樣板開發的觀念，來增加開發上的效率。在 3.1 節中將說明視覺化的軟體建構模型 。在 3.2 節中將說明視覺化的軟體建構模型的細部流程。

3.1 視覺化軟體建構模型

由於在傳統的使用者介面開發方式上有上述第二章所討論到的問題，爲了解決此問題，有另一種以不同角度切入，視覺化需求爲導向的軟體建構模型。[13]

在傳統方法中的實作階段，如圖 12 所示，由程式設計師來依照所有需求所訂定下來的規格將系統實作出來，而在使用者介面實作上也是由程式設計師來實作。而使用者介面設計者在實作階段只是作爲協助，製作使用者介面所需的媒體檔案、資源檔案，如圖示、背景圖、聲音檔等。

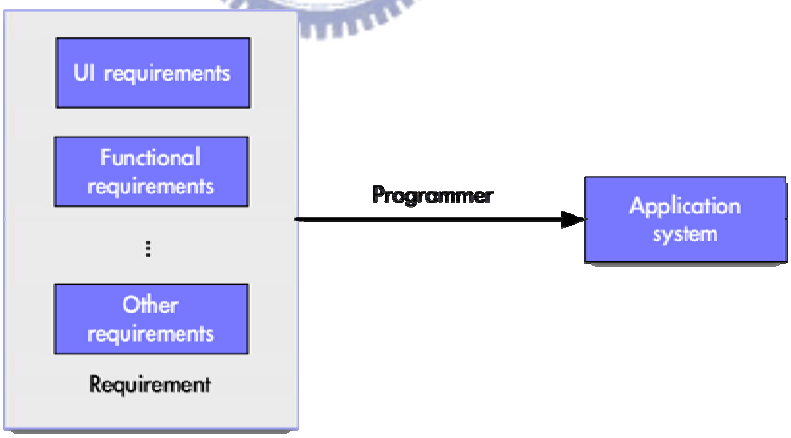

圖 12 傳統由軟體開發者開發方式

而視覺化的軟體建構模型將實作的工作分開爲兩部份：

- 使用者介面實作

- 使用者介面相關的功能及其它系統部分實作

使用者介面設計者負責設計實作使用者介面，而程式設計師負責實作與使用者介面相關的功能及其它系統部分，之後使用者介面設計者將兩部分加以連結，產生最終的系統，如圖 13 所示。[4][13][15]

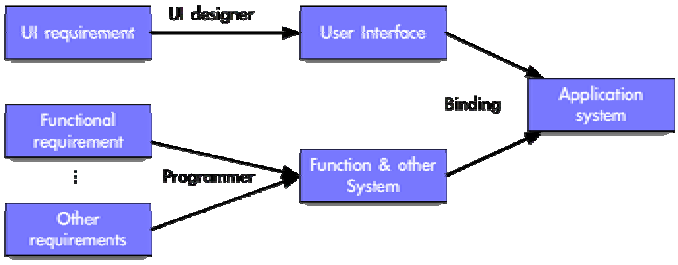

圖 13 分別由程式設計師和使用者介面設計者的開發方式

此模型主要分爲兩個部份，分別爲視覺化的需求編輯和程式碼產生器，而爲了將此模型適用於開發行動電話的使用者介面，我們做了部分的修改與延伸，且爲了加速開發使用者介面的過程，在此模型額外加上了樣板資料庫的功能。

在開發各個使用者介面時，其使用者介面會有相似的地方，因此我們取出相同的地方，成爲樣板，儲存在資料庫中，來供日後開發時重複使用，在第四章會詳細說明樣板的由來。

在視覺化的需求編輯部分，以 UI Visual Requirement Authoring System 爲中心，透過它將 Visual Requirement Representation[18]產生出來，即我們要的使用者介面。而程式產生器以 Program generator 爲中心將 Visual Requirement Representation 轉換爲目標平台的程式。

一個 Visual Requirement Representation 是以視覺化的方式將使用者介面呈現出來，如圖 14 所示，它由以下的元素所構成:

- 演員 (Actor) : 又稱之爲 MRC (Multimedia Reusable Component) 爲構成此 Visual Requirement Presentation 中最基本的單位，他以物件導向的方法將多媒體資料封裝起來，擁有自己的屬性，並能將受外部的訊息來觸發自己的動作。

- 劇情 (Scenario) : 劇情是用來描述 Actor 在場景中時間與空間上的關係。
- 場景 (Scene) : 用來放置 Actor，在每個場景中皆有一份劇情。

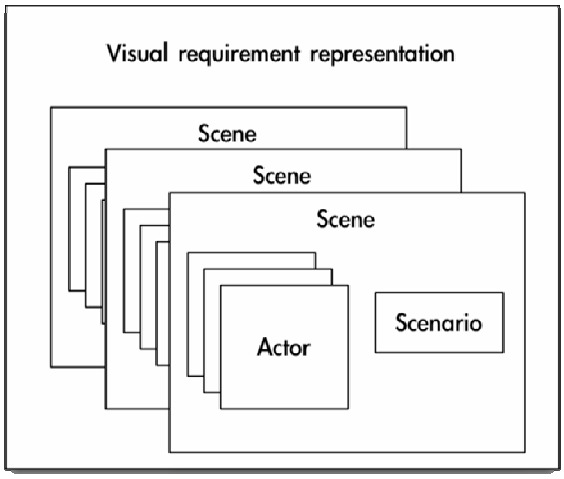

圖 14 Visual requirement representation

圖 15 爲延伸的視覺化軟體建構模型，各部分詳細說明如下：

- UI designer: 利用 UI Visual Requirement Authoring System 依照使用者介面設計的需求，將使用者介面開發出來。
- Component constructor：元件製作工具，當需要製作新的 MRC 時，則利用此工具來製作，完成後存入 MRCs Manager 供 UI Visual Requirement Authoring System 日後再使用。
- MRCs Manager: 用來儲存各個 MRC。
- UI Visual Requirement Authoring System: 以描述語言爲基礎的視覺化的編輯環境，使用者可以在不寫程式而完成一份多媒體呈現的製作。且可利用樣板資料庫，來產生基本的使用者介面，接下來做進一步的修改，節省製作的時間。
- UI Template Constructor: 負責製作樣板，將樣板存入資料庫中。
- Template Database: 存放 UI Template Constructor 所設計 的樣本，來供 UI Visual Requirement Authoring System 來使用。
- Target UI Requirements Presentation: 經過 UI Visual Requirement Authoring System 的設計後，我們可以得到一分滿足 UI Requirement 的使用者介面。
- Program generator: 將使用者介面與所需要的各個功能連結起來，再將 Target UI Requirements Presentation 轉換爲目標平台的程式碼。

- Design framework: 基礎的程式元件，如 User Interface framework，提供使用者介面元件，這些程式元件爲 Program generator 在產生程式所需要的。
- Function library component: 由程式設計師依照硬體功能所設計的函式庫，供 Program generator 來使用。
- Generated Application Software System: 經由上述的系統，最後產生的目標平台的程式。

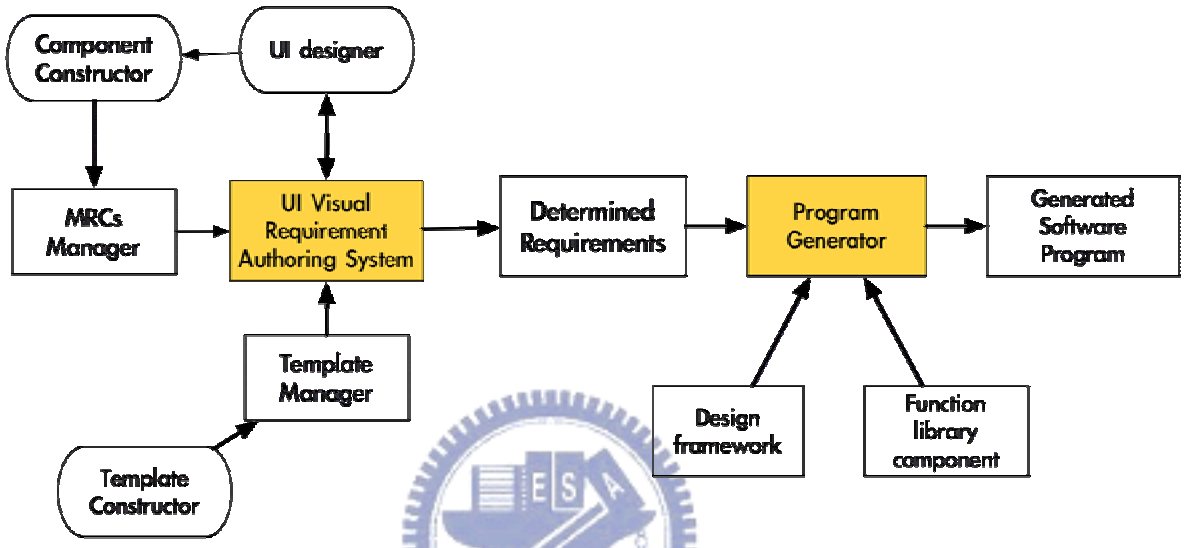

圖 15 視覺化軟體建構模型

此模型使用步驟如下:

1. 使用者介面設計者透過 UI Visual Requirement Authoring System 的協助，利用現成的 MRC，依照使用者介面視覺化需求，將使用者介面設計出來。
2. 若不存在合適的 MRC，則使用者介面設計者利用元件製作工具將新的 MRC 設計出來。
3. 當新的 MRC 製作出來，則存入元件管理系統內 (MRCs Manager)，來供 UI Visual Requirement Authoring System 使用。
4. 若要製作的使用者介面在樣板資料庫中有類似的，則可以直接從樣板資料庫內產生出來，交由 UI Visual Requirement Authoring System 進一步編輯。
5. 當編輯動作完成，則由 UI Visual Requirement Authoring System 產生符合要求的 visual requirement representation。
6. 將 visual requirement representation 與適當的功能與連結起來，直到滿足功能上的需求爲止 。
7. 最終產生出目標平台的程式碼。

3.3 視覺化軟體建構方法流程

依照這個模型，規畫了以下的細部開發流程，分爲三個階段:

1. 使用者介面需求編輯階段: 屬於在此模型內的 UI Visual Requirement Authoring System 部分
2. 產生目的碼階段: 屬於在此模型內的 Program generator 部分
3. 模擬階段: 屬於在此模型內的 Generated Application Software System 部分

以下做進一步的說明。圖 16 爲細部的流程圖

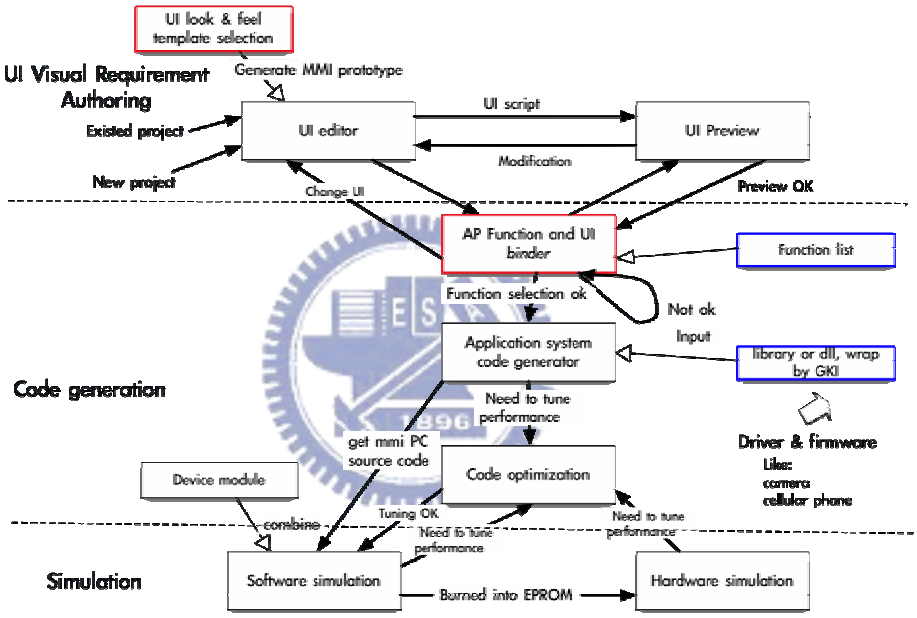

圖 16 視覺化軟體建構方法流程圖

3.3.1 使用者介面需求編輯

首先是使用者介面需求編輯，在此階段由二個步驟所構成，分別是

- UI editor: 使用者介面設計者可以重新建立或載入之前設計的使用者介面，如果重新建立，可以自行將使用者介面產生，或由樣板資料庫中挑選，將使用者介面的雛型產生出來，接下來做進一步的編輯修改，如加入新的文字或圖片。

- UI preview: 若使用者介面設計者想要觀看編輯後實際運作的結果，則可進入此步驟，在此模式，所操作的畫面，會如同在實際機器執行的畫面。

3.3.2 產生目的碼

當使用者介面需求編輯後，接下來是產生目標平台的程式碼，在此階段由三個步驟所構成，分別是

- AP function and UI binder: 當設計出的使用者介面符合需求時，則進入功能連結步驟，選擇要使用的硬體，此選擇會影響所能連結的函式種類，例如沒有選擇使用相機模組，則與相對應的拍照，儲存照片等功能則不會出現供使用者介面設計者選擇連結。於是使用者介面設計者將所需要的功能，與使用者介面與做連結。

- Application system code generator: 當使用者介面完成與所有所需要的功能連結之後，則進入此步驟，將之前編輯動作系統所儲存的描述檔轉換爲目標平台的程式碼。

- Code optimization: 若程式執行時的效率不佳，則進入此步驟，調整程式碼來達到最佳化。

3.3.3 模擬階段

完成上一個階段後，我們得到了目標平台的程式碼，最後進入模擬階段，在此階段由二個步驟所構成，分別是

- Software simulation: 因爲將程式下載到實際的機器上需要時間，且在桌上型電腦較方便除錯，因此轉換出來的程式碼先進入此步驟，在桌上型電腦上與模擬器及搭配各個所需的硬體一起執行，如相機模組，驗證功能上是否正常，使用者介面運作情形是否如當初所想。

- Hardware simulation: 在模擬器上執行無誤後，則進入此步驟將程式下載至 EPROM 上，在實際上的硬體上執行。

四、行動電話之使用者介面設計

在第三章我們討論了以視覺化軟體建構方法的整個開發方法與流程，在這一章我們將要討論如何將視覺化軟體建構方法用於行動電話的使用者介面的設計上，且如何由各個使用者介面中得到樣板。在 4.1 中將討論如何將視覺化軟體建構方法用於行動電話的使用者介面的設計上。4.2 節中將討論樣板的設計。在 4.3 節中將會說明如何應用樣板在設計行動電話使用者介面上。

4.1 行動電話使用者介面

操作行動電話時，我們首先看到的是待機畫面，在這個畫面顯示了行動電話相關的狀態，而按下數字鍵時，則跳出撥號的畫面，供使用者輸入號碼來撥號，或在畫面上選擇一功能，按下功能鍵時，則跳入相對應的畫面或執行相對應的功能。

於是我們可將整個使用者介面分解，以頁面的方式來顯示，每個頁面代表著一個操作畫面，而當選擇功能跳入其他頁面的動作或者是執行某功能，可以以連結的方式來代表。當我們以頁面方式來描述一個完整的使用者介面時，則得到一個樹狀結構，如圖 17 所示，此結構由二項成員組成:

- 節點(Node): 畫面或者是功能
- 連結(Link): 畫面與畫面或是畫面與功能之間的關係

因此，我們能以這樣的樹狀結構將使用者介面記錄下來，接下來，我們進一步討論節點內的畫面與功能的設計。

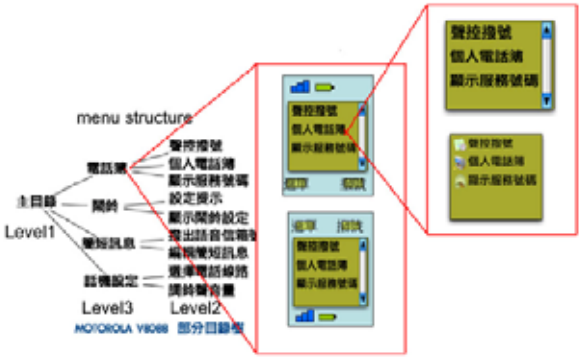

圖 17 Motorola V8088 部分目錄樹

4.1.1 畫面

觀察使用者介面的各個畫面，可以發現，畫面是文字和圖形所構成，那要如何方便使用者介面設計者設計出使用者介面呢，我們先觀察在桌上型電腦上使用者介面設計方式。

在桌上型電腦上的使用者介面開發，如 Borland C++ Builder, Visual Basic 等，提供了基本的使用者介面元件來使用，使用者可使用這些基本的使用者介面元件來設計出複雜的使用者介面，如此使用者介面設計者可以不用花費時間在以程式開發使用者介面上。

同樣的將此觀念運用在行動電話上，因此需要提供使用者介面元件來供使用者介面設計者來使用，於是依照所觀察到的畫面，歸納了以下所需要的使用者介面元件

- 文字演員：提供顯示文字的功能
- 圖片演員：提供顯示圖片的功能
- 輸入盒演員：提供使用者介面在實際執行時，能提供使用者作輸入
- 列表盒演員：提供當多列資料的顯示功能，

既然有了這些元件，我們需要有一個視覺化的環境來編輯它，而本實驗室之前開發的視覺化編輯器，恰好可以提供視覺化的環境來設計行動電話使用者介面，於是我們便以此編輯環境爲基礎加以延伸。[4][8][10]

此視覺化編輯器，所編輯出的內容是以一個個場景，所有的場景集合起來，稱之爲一個故事 (Story)，每一個場景 (Scene) 都爲一個畫面，而畫面是由演員所構成 (Actor) ，各演員都有各自的動作，使用者可以任意的編輯來完成多媒體的呈現 (Visual requirement representation)。

因此將視覺化編輯器與行動電話的使用者介面做以下的對應：

其關係如下:

- 演員 (Actor)：爲使用者介面元件，構成畫面的最基本單位，使用者介面元件可連結到某一特定的功能。
- 場景 (Scene)：爲各個節點的使用者介面，顯示各個使用者介面元件。

- 故事 (Story)：由各個選單構成，以連結 (link)，連結各個節點。

有了以上的對應關係，因此我們就能利用以此視覺化編輯器來編輯使用者介面，而此編輯器對應到視覺化軟體建構模型的 UI Visual Requirement Authoring System。

4.1.2 功能

在 3.3.2 所提到的功能連結步驟 (AP function and UI binder)所能連結的功能，是由實作的函式庫所提供，這函式庫由程式設計師來實作。並依硬體的種類加以分類整理，提供相對應的列表在功能連結時使用。[4]

那在手機中的功能有那些呢？在觀察 Motorola v8088, Sanyo J88, Ericsson T-39 等數款行動電話之後，我們大致上將功能分爲兩大類：

- 基本功能：行動電話上必需具備的功能
- 加值和硬體相依功能：不爲行動電話所必備，但此功能可以增加消費者使用上的便利性或產品上的吸引度。

而這兩大類又可細分爲以下的項目，列表如下:

表 2 行動電話所需功能表

基本功能	加值和硬體相依功能
通話功能	遊戲
電話簿	錄音
信息	照像
通話記錄	Java
通信設定	Video player
環境設定	Audio player
	紅外線
	藍芽

基本功能詳細說明如下:

- 通話功能: 提供撥接，掛斷電話等功能。
- 電話簿: 管理聯絡人名單，方便查詢。
- 信息: 提供收發簡訊，多媒體簡訊，e-mail 等功能。
- 通話記錄: 記錄通話時間，通話費用，來電號碼等功能。
- 通信設定: 與系統商相關的設定，如來電轉接，撥號顯示來電等功能。
- 環境設定: 與行動電話本身相關的設定，如來電鈴聲，鈴聲音量等，螢幕亮度等功能。

而加值和硬體相依功能方面並不固定，是隨著使用者需求與流行不斷的作變動，但以上列表爲目前新款行動電話可能具備的功能，詳細說明如下。

- 遊戲: 提供小遊戲供消費者遊玩，打發時間用，在遊戲的開發上，有部份改以 Java 來開發，能在大部分支援 Java Virtual Machine 上的行動電話上執行。
- 錄音: 提供錄製電話語音，語音備忘錄等功能。
- 照像: 提供拍照功能，更進一步可提供錄影方面的功能。
- Java: 提供 Java Virtual Machine，讓以 Java 程式可執行。
- Audio player: 提供撥放器來撥放音樂，如撥放 mp3, midi, wma 的音樂格式。
- Video player: 提供撥放器來撥放影片，如撥放 mpeg4, mpeg2, wmv 的影片格式。
- 紅外線: 提供以紅外線的方式與桌上型電腦，或行動電話之間的資料傳輸。
- 藍芽: 提供以藍芽 (Blue tooth)方式與其他裝置做資料的傳送，如藍芽耳機，桌上型電腦等。

而程式設計師必須將這些分類的功能實作出來，使用者介面設計者才能在功能連結步驟中，使用這些功能。

然而在硬體演進的同時，然是相同功能的硬體，但是函式庫也需隨著做更換，這樣的更換也會沖擊到使用者介面，使用者介面必須做相對應的調整，爲了減少這樣修改的工作，於在使用者介面與其它系統之間加了一層介面，稱之爲 GKI(Generic Kernel Interface)，此部分的設計爲劉柏昌同學負責設計與實作。

使用者介面要和其它系統彼此溝通都需要透過此介面。不管更換了什麼硬體，更換了什麼作業系統，只要實作此介面，使用者介面便可以達到最少的修改來適應新的系統 。

4.2 樣板

在比較各個行動電話的使用者介面時，可以觀察到雖然是不同機型，但爲同一家公司出的機器，其使用者介面外觀有類似的地方，或者是雖然是不同公司出的機器，但是整個使用者介面結構相似，其差異在於使用者介面的外觀上。

而當我們開發這些相似的使用者介面時，我們要重新製作嗎？ 如果我們能夠將重複的地方抽取出來，日後在編輯相似的使用者介面時，直接加以利用，減少重複的 開發工作，這樣的構想稱之爲樣板。[5]

由 4.1 節看到，完整的使用者介面以各個場景所構成，而各個場景由各演員所構成，而演員的內部屬性，影響了使用者介面的外觀。可以 MVC 模式 (Model, view, control) 將使用者介面分爲兩部份，有關使用者介面結構部分的稱之爲 Model，而場景的外觀部分稱之爲 View。

而這兩個部份可以作以下的組合:

- Model 不變 View 變動
- View 變 Model 不變動

而要如何讓 Model 與 View 可以在彼此互換下依舊可以正常運作呢？在 HTML 中有一個標準稱爲 CSS (Cascading style sheets) ，由 W3C (World wide web consortium) 所提出，透過 CSS，可以設定 HTML 的文件外觀，如文字顏色，大小等。[14]

在傳統的 HTML 中，若要在標題定義一個文字的顏色，需在標題標籤中插入 Font 標籤，顏色爲白色，如:

```
<H3><FONT COLOR= “while”> Title </FONT></H3>
```

若有 10 個標題要更改，則需要在 10 個標題中插入 Font 標籤。若有一天想把所有標題更改成其他顏色，則需要一個個的修改。

若改爲 CSS 來處理的話，只需先定義:

H3 { color: while; }

之後只要使用 H3 標籤的標題就會全部都爲白色，一但需要更改所有的標題，修改 CSS H3 標籤定義的部分即可。

基於以上的方式，將 Model, View 的兩部份分開，分別存爲樣板，當要產生可編輯的使用者介面時，則將此兩部分組合，成完整的使用者介面。

而這樣帶來的好處是不同的 Model 與 View 的搭配即能產生不同的使用者介面，並只要定義少數幾種樣式，即能對整個使用者介面做更動。

而 View 的部分在進一步的來看，在頁面中是由各個演員所構成，它們在畫面上的排列的方式稱之爲版面，而它們所呈現出來的畫面稱之爲樣式。

因此由 data 與 view 中，定義了以下的三種樣板:

- 結構樣板 (Structure template) : 使用者介面目錄結構。
- 版面樣板 (Layout template) : 呈現排列方式。
- 樣式樣板 (Style template) : 樣式外觀。

接下來分別一一做討論

4.2.1 結構樣板

在 4.1 節中的討論，我們以節點，與連結的關係來描述樹狀結構，更進一步的我們可以用此關係來描述任何的拓撲 (topology) 形狀，於是可能會有下的結構，如樹狀，列狀，環狀 。如圖 18。

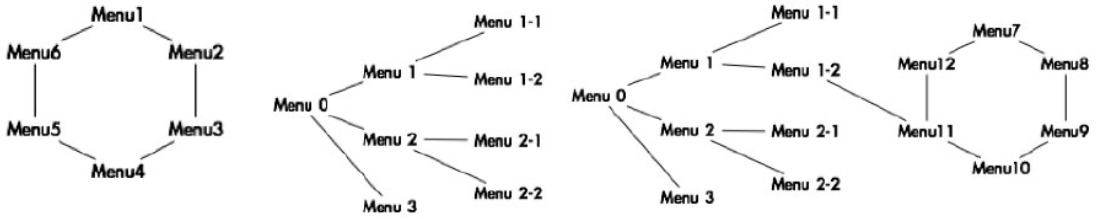

圖 18 各種使用者介面結構

結構樣板中存放了整個使用者介面結構資料，包括每個場景中的演員，所使用的版面與樣式名稱，而版面與樣式的細部內容則定義在以下的二種樣板。

於是結構樣板由下列兩項資料構成:

- 整個故事的結構
- 場景中的演員資料。

4.2.2 版面樣板

在每個場景中，畫面是由各演員擺放位置來決定，這種排列方式稱之爲版面， 於是依照版面的資訊來定義版面樣板 。依照版面樣板的不同更改各演員 擺放的位置。例如一種版面是文字在上，圖示在下，而另一種版面可能是文字在下，圖示在上。 如圖 19 所示：

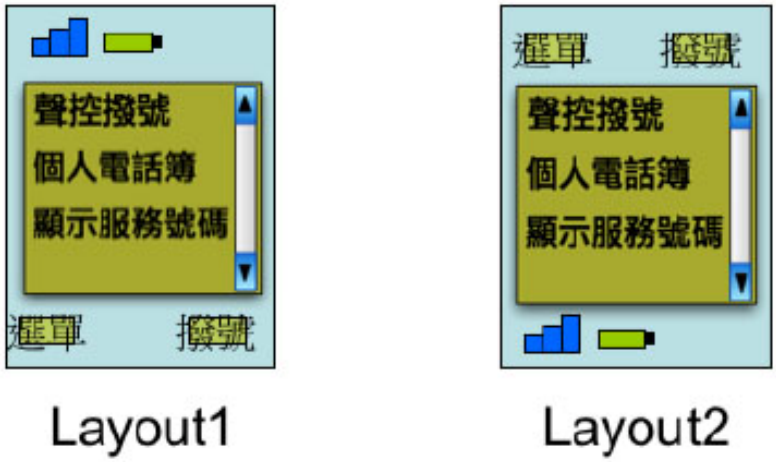

圖 19 不同的兩種版面

而在一個完整的使用者介面中，其擺放位置是有規律性的，以樹狀結構的使用者介面來說，相同層數的場景版面大多是相同的，因此只要定義少數的版面即能描述完整使用者介面內的各個場景的版面。因此將使用者介面會用到的版面放在同一個版面樣板中，抽換即可更換整個使用者介面的版面。

版面樣板由各個場景的版面資料所構成，而場景的版面由下列兩項資料構成

- 版面名稱
- 各個演員在此版面的資訊

4.2.3 樣式樣板

在每個場景中，演員所呈現的外觀是由本身的屬性來決定的，即使是相同的演員，會因爲屬性的不同而有不相同的外觀，如圖 20 的例子， 由文字爲主的外觀藉由更改屬性成爲以圖示爲主的外觀。於是將與外觀有關的屬性稱之爲樣式獨立出來成爲樣式樣板。而在一個完整的使用者介面中，其各個樣式是有規律性的，例如右下角的文字顏色都是黑色。因此只要定義少數的樣式即能描述完整使用者介面內各個演員。因此將使用者介面會用到的樣式放在同一個樣式樣板中，抽換即可更換整個使用者介面的樣式。

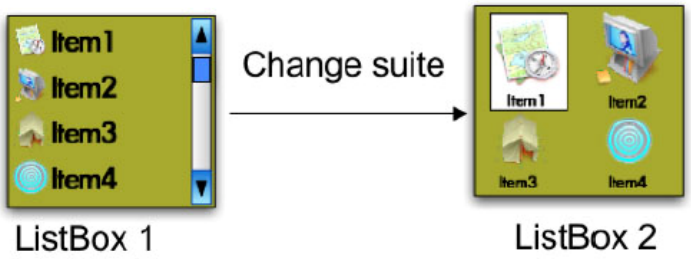

圖 20 兩種不一樣的樣式

樣式樣板由各個演員的樣式資料所構成，而演員的樣式資料由下列一項資料構成

- 與演員外觀相關屬性資料。

4.2.4 行動電話之泛型使用者介面樣板

將上述的討論的三種使用者介面樣板，結構樣板，版面樣板與樣式樣板結合在一起，我們可以得到一個通用型的樣板，稱之爲行動電話之泛型使用者介面樣板(Generic UI Template for handset devices)。

由結構樣板產生 Model，而版面樣板，樣式樣本提供 View，透過三種樣板的搭配組合， 能夠產生不同的使用者介面， 如圖 21， 結構樣板產生樹狀的結構，再由版面樣板提供演員在場景中的版面資訊，最後再由樣式樣板來提供各個演員的樣式，最後得到使用者介面的雛型。

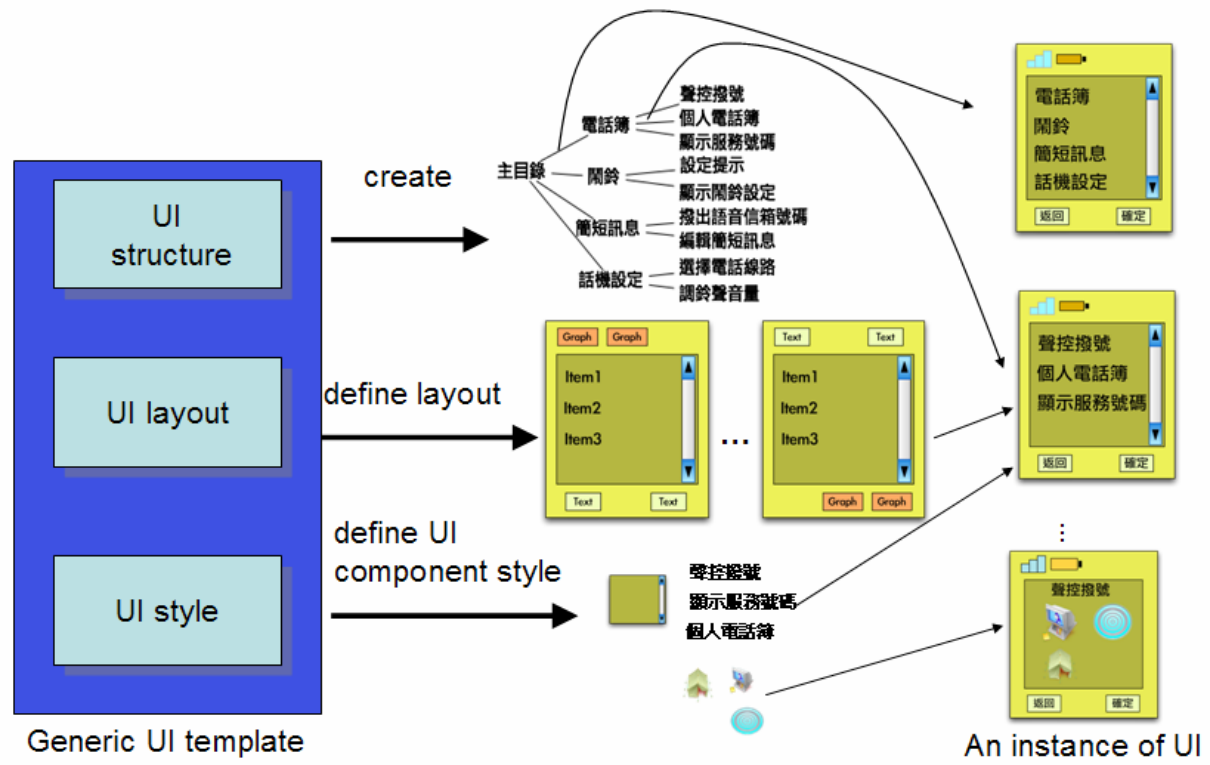

圖 21 行動電話之泛型使用者介面樣板

若這三種樣板各有 a , b, c 個樣板，則會有 a x b x c 種組合

4.3 以樣板來建立使用者介面之步驟

在 4.2 討論了行動電話之泛型使用者介面樣板的產生，接下來說明如何應用它來設計行動電話的使用者介面，使用時分爲五個步驟，我們隨著步驟舉例做爲說明：

1. 選擇一個泛型使用者介面樣板

由使用者選擇所要搭配的結構樣板，版面樣板，樣式樣板的種類，將選擇輸入系統。

2. 從所選擇的結構樣板中產生整個使用者介面結構

接著系統依照使用者所選擇的結構樣板，建立所有的場景及場景中的演員，且建立視覺化編輯器所需要的檔案，如描述檔，媒體檔案。

如圖 22，我們選擇了樹狀的結構樣板，於是系統從主目錄開始，依序將所有的場景及場景中的演員及每個場景所需要的檔案建立起來。

圖 22 使用者介面之樹狀結構

3. 對從第二步驟所產生出的場景，套用相對應的版面

對每個場景，系統依照結構樣板的資訊，至使用者所選擇的版面樣板內，取得相版面資訊，填入場景中。

如圖 23，在電話簿這層的場景中，系統依照結構樣板的資訊，從版面樣板中挑選了以文字的擺放方式版面，決定了此場景中，各個演員的擺放位置。

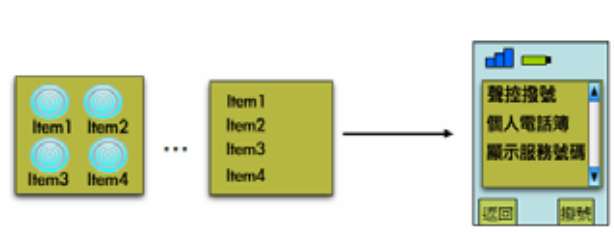

圖 23 版面選擇

4. 對從第二步驟所產生出的演員，套用相對應的樣式

對每個演員，系統依照再結構樣板的資訊，至使用者所選擇的樣式樣板內，取得相對應的資訊。

如圖 24，系統依照再結構樣板的資訊，至樣式樣板內，挑選了以圖示爲主的樣式，決定了此場景中的演員樣式。

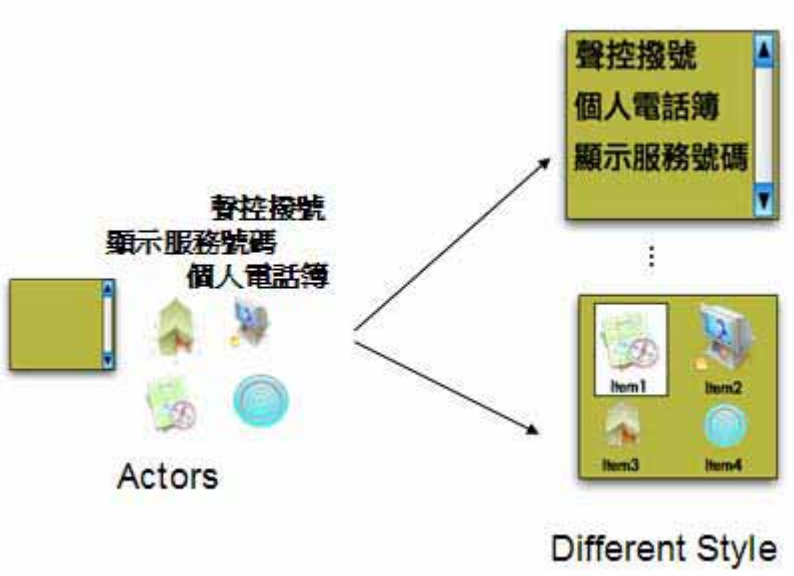

圖 24 樣式選擇

5. 重複第三與第四步驟直到所選擇的結構樣板裡的所有場景和演員都被套用完成

系統不斷的重複著第 3 與第 4 步驟，直到使用者介面所需要的版面與樣式資訊，皆填入使用者介面爲止。

如圖 25，完成了電話簿部分的場景設定，接下來系統繼續重複第 3 與第 4 步驟，直到完成樹狀結構使用者介面所有的場景。

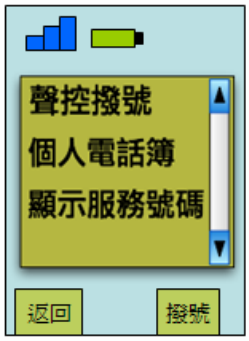

圖 25 由手機之泛型使用者介面樣板所產生的電話簿的使用者介面

在以樣板產生使用者介面之後，我們可以得到一個使用者介面的雛型，供使用者進一步做編輯，接下來再進行使用者介面功能連結，最後產生目的碼，編譯之後即可在模擬器上執行。

五、系統設計與實作

在本章中將要討論樣板的設計與實作。在 5.1 節中將會討論樣板系統的架構。5.2 節中將討論樣板的實作、所用的資料結構。5.3 節中將討論樣板系統的運作流程。

5.1 樣板管理器

由前一章的討論，設計出了行動電話之泛型使用者介面樣本，爲了使用這些樣板我們需要實作樣板管理器，負責將這些樣板組合，產生可以編輯的使用者介面。

這三個種類的樣板都有各自的管理者，而這三個樣板的管理者，再由一個總管理者來負責管理。

系統部分以下子系統，分爲 9 部分

1. TemplateManager: 管理 Structre, Layout, Style 三種 TemplateManager
2. StructreTemplateManager： 管理 Structure Template
3. LayoutTemplateManager： 管理 Layout Template
4. StyleTemplateManager： 管理 Style Template
5. StructureTemplateBean： StructureTemplate 之 instance
6. LayoutTemplateBean： 各 LayoutTemplate 之 instance
7. ActorLayout： 存放各 Actor layout 之資訊
8. StyleTemplateBean： 各 StyleTemplate 之 instance
9. ActorStyle： 存放各 Actor style 之資訊

圖 26 爲樣板管理器的架構圖

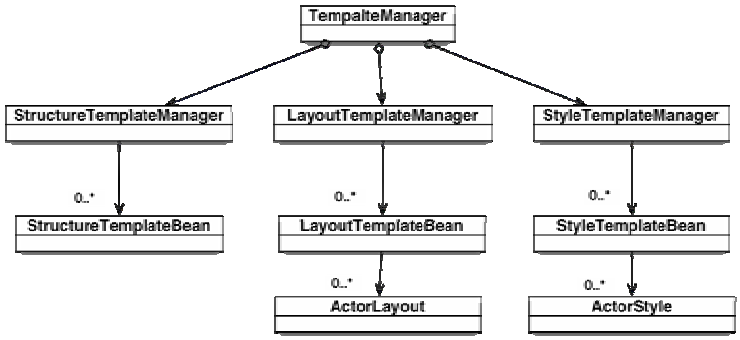

圖 26 樣板管理器的架構

5.2 行動電話之泛型使用者介面實作

在實作樣板時，爲了程式使用上的方便，在版面樣板與樣式樣板設計上使用了 design pattern [12]中的 prototype 來設計。

在生成版面樣板與樣式樣板時，便能不用考慮他是屬於那一種版面或那一種樣式，只要使用運用它們本身實作的 clone method，便能生成新的樣板。

而在樣板的資料儲存方面，爲了處理上的方便，使用了 XML 的資料格式。

圖 27 爲版面和樣式的類別圖。

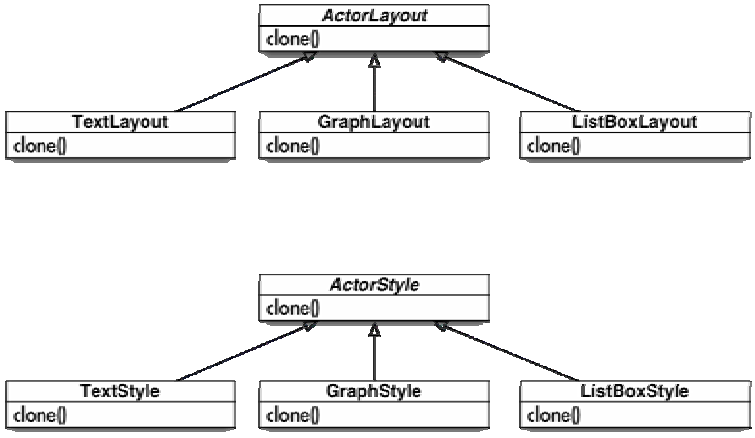

圖 27 Layout 與 Style 類別圖

5.2.1 結構樣板之實作

結構樣板類別之內容摘錄如下:

```
Class StructureTemplateBean {
   vector<SceneNode> mNodes;
   map<int, vector<TMCActor *> > mSceneActor;
}
```

一個結構樣板中，以 SeneNode 的 mNodes 記錄了整個使用者介面的結構，一個 SceneNode 中主要記錄了以下的資料

- 場景的 ID: 用來代表此場景的代號，此代號不能重複;
- Scene 子節點 ID: 用來記錄所連結的結點;
- Scene 用的版面名稱: 對應到版面樣板的資料。

而至於各場景中的演員資料，則以 STL (Stand Template Library) 的 map 結構來記錄，以各個場景的 ID 爲索引。

而結構樣板之儲存在 xml 資料結構如下

它分爲 4 部分:

1. SystemInfo: 說明所使用的 Script 版本
2. TemplateInfo: 說明此樣板的資訊
3. Structure: 描述整個結構資訊，裡面記錄了節點的名稱，所使用的此節點使用的版面名稱，他的子節點。

例如下面的例子:

```
<Scene ID="0" Name="待機" Filename="@sc223"
Layoutname="level_0">
        <ChildScenes>
          <ChildScene ID="1"/>
          <ChildScene ID="2"/>
   </ChildScenes>
</Scene>
```

此場景的 ID 爲 0, 場景的名稱爲待機，描述檔檔案名爲 @sc233，所使用的版面名稱爲 level_0, 它有兩個子節點，ID 分別爲 1 和 2。

4. Scenes: 記錄了各個場景中裡的演員資訊

例如下面的例子:

```
<Scene ID="1" BackgroundPic="@sc287.jpg">
        <Actors>
          <Actor Type="MCText" Name="Actor1" Layout="text1"
Style="textstyle1" Text="選擇"/>
        </Actors>
</Scene>
```

此場景 ID 爲 1 ，背景圖檔爲@sc287.jpg，裡面有一個演員，種類爲 MCText，稱爲 Actor1，所使用的版面爲 text1，所使用的樣式爲 textstyle1，他的屬性 Text 內容爲 選擇。

5.2.2 版面樣板之實作

版面樣板類別之內容摘錄如下：

```
Class LayoutTemplateBean{
      map< String, map<String, ActorLayout> > mLayoutList
};

Class ActorLayout{
   clone();
}
Class TextLayout : public ActorLayout{}
Class GraphLayout : public ActorLayout{}
```

一個版面樣板中，以 mLayoutList 記錄了一整組使用者介面所用的版面，而一個版面，包含了此版面各個的版面資訊。

版面樣板之儲存資料結構如下

它分爲 3 部分:

1. SystemInfo: 說明所使用的 Script 版本

2. TemplateInfo: 說明此樣板的資訊
3. Scene: 說明各 Actor 在版面擺放的位置

```
<Scenes>
        <Scene ID="0" Layoutname="level_0">
            <Actors>
                <Actor Type="MCText" Layout="text1" Left="70"
Top="60"/>
            </Actors>
        </Scene>
<Scenes>
```

有一稱之爲 level_0 的版面，裡有一演員，版面的名稱爲 text1，其擺放位置爲 left 爲 70，top 爲 60

5.2.3 樣式樣板之實作

樣式樣板類別之內容摘錄如下:

```
Class StyleTemplateBean{
  map< String, ActorStyle* > mActorStyleList;
}

Class ActorStyle{
   clone();
}
Class TextStyle : public ActorStyle{}
Class GraphStyle : public ActorStyle{}
```

一個樣式樣板中，以 mActorStyleList 記錄了一整組 UI 的樣式， 而一個樣式，包含了此演員的樣式資訊。

樣式樣板之儲存資料結構如下

它分爲 4 部分:

1. SystemInfo: 說明所使用的 Script 版本

2. TemplateInfo: 說明此樣板的資訊
3. Actors: 說明各演員的一些屬性參數，不同種類的演員會有不同的參數

```
<Actors>
        <Actor ID="0" StyleName="textstyle1" Type="MCText"
FontColor="0"/>
</Actors>
```

有一演員的樣式稱爲 textstyle1，他的字型顏色爲 0(黑色)

5.3 樣板管理器之運作流程

在使用樣板管理器時，首先使用者先選擇要使用的結構，版面，樣式三種樣板，接下來樣板管理器透過此三種子樣板管理器將選擇的樣板挑選出來。

接下來產生整個使用者介面，先由結構樣板產生整個使用者介面的結構，接下來對每個場景中的演員，查出所使用的版面名稱，至版面樣板內查出相對應的版面資訊，填入演員內，再至樣式樣板內查出相對應的樣式資訊，填入演員內，直到所有場景中的演員都尋訪過。下圖 28 爲流程圖

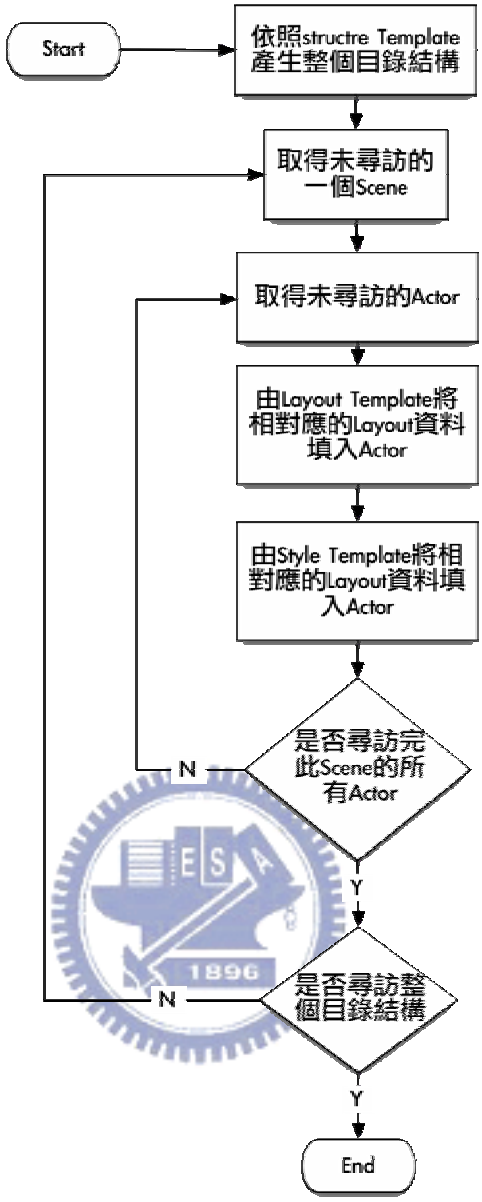

圖 28 樣板管理器之運作流程圖